**About the software**

For pleasant-looking LED animations a little programming effort is required. Simply blinking the LEDs in a random fashion is not enough.

**Refresh rate**

The display consists of a 6-by-6 LED matrix. To keep things reasonably simple we decided to use a multiplexing technique where only one LED is lighted at any time. To achieve a nice, flicker-free display the refresh rate should be at least 100 Hz which when having 36 LEDs corresponds to 3600 Hz. We used 5 kHz, resulting in a refresh rate of almost 140 Hz. Timer 0 is used for this task, we called it the systick timer (system tick).

From this timer we also derived a millisecond and a second timer.

**Brightness**

It is possible to control LED brightness by switching the active LED off before the next multiplexing cycle. Maximum brightness is obtained when the LED remains on for the full multiplexing cycle, minimum brightness is achieved when the LED is switched off as soon as possible after it was switched on.

A timer is a good way to control the brightness. The timer is started when the LED is activated and when it expires it deactivates the LED. Timer 2 was used for this. It is started by the systick timer every time the latter updates the LED matrix. With a clock frequency of 16 MHz and a prescaler value of 32 we can control the brightness up to 510 µs (8-bit value) in steps of 2 µs. In reality it will never be longer that the systick rate allows it to be (200 µs), because the systick timer will restart the brightness timer before it can expire.

**Multitasking**

For complex animations it is useful to have some sort of multitasking mechanism at hand that allows multiple simple animations to use the same display without interfering with each other. The difficulty is to share an LED between several tasks. If one tasks wants it on while another wants it off, how do you manage that? By keeping track of LED usage. Every time an animation thread wants to activate an LED, the LEDs usage counter is increased. When a thread wants to deactivate the LED, its usage count is decremented. Only when the usage counter is equal to zero will an LED be switched off. This technique gives precedence to the on-state. The result is smooth animations where LEDs can run around freely without being switched off in unpredictable ways.

**Animations**

We settled on two types of animations that run through each other. The first type consist of periodically switched-on rings. Which ring (1 out of 6) is decided by a random number generator. Every LED in the ring is assigned a random timeout period after which it will be switched off. What you will see is a ring switching on and then slowly degenerate when its LEDs one after the other will “die”.

The second type of animation is the so-called “running led”. This consists of an LED that follows a predefined path with programmable speed, direction, acceleration, deceleration and starting delay. With only one running led you will see an LED spiraling clockwise or anti-clockwise from the outside (LED1) to the inside (LED36). It will start slowly, then accelerate to reach its maximum speed, continue for a while before decelerating and finally reaching the last LED. Every parameter is programmable and can be changed on-the-fly. We used six running LEDs.

To spice things up a bit brightness modulation was added giving it a kind of heartbeat. Note that natural brightness control requires exponential values, which explains why we used shifting bit values (which equals to multiplying by 2) instead of a simple counter.

**Project**

The project is an Atmel Studio 6.2 GCC project without fancy ASF (Atmel Software Foundation) stuff. The functions are spread out over several files with meaningful names and commented where needed so that it should not be too difficult for the interested user to modify the animation.

Fuse settings are described in the main file, for sake of completeness here they are:

EXTENDED = 0xFD

HIGH = 0xDF

LOW = 0xE6

The executable HEX file can be used with the ATmega328 and ATmega328P.

- CPV -